Course Number & Name: CSC 122 Computer Science II

Credit Hours: 4.0  Contact Hours: 4.0  Lecture: 4.0  Lab: N/A  Other: N/A

Prerequisites: Grades of “C” or better in CSC 121 and MTH 113 or MTH 119

Co-requisites: None  Concurrent Courses: MTH 113 or MTH 119

Course Outline Revision Date: Fall 2010

Course Description: This course explores further the concepts introduced in CSC 121, applying them to more complex problems. Areas covered include class construction, class instantiation, file/stream processing, list processing, string processing, dynamic storage allocation, and internal search/sort methods.

Course Goals: Upon successful completion of this course, students should be able to do the following:

1. demonstrate knowledge of basic concepts and methodologies of computer science;
2. design application programs to implement algorithms in an object-oriented language;
3. apply the principles of software design to a set of selected applications;
4. use a computer system as a tool for problem solving; and
5. communicate using accurate computing terminology and notation in written and/or oral form.

Measurable Course Performance Objectives (MPOs): Upon successful completion of this course, students should specifically be able to do the following:

1. Demonstrate knowledge of basic concepts and methodologies of computer science:
   1.1 define and discuss relevant ‘computer science’ words and concepts;
   1.2 explain the relation of the ‘computer science’ concepts to computing; and
   1.3 use computer science methodologies to solve problems

2. Design application programs to implement algorithms in an object-oriented language:
   2.1 discuss and apply the basic structures of programming language; and
   2.2 define ‘class’ and use ‘classes’ to implement an ADT
Measurable Course Performance Objectives (MPOs) (continued):

3. Apply the principles of software design to a set of selected applications:
   3.1 document the process of movement from problem to solution; and
   3.2 create the source code that implements the solutions

4. Use a computer system as a tool for problem solving:
   4.1 using an integrated programming environment, write and complete a program; and
   4.2 execute and test the solution to the problem

5. Communicate accurate computing terminology and notation in written and/or oral form:
   5.1 document the programs so they are self explanatory; and
   5.2 use correct computing terminology on written and oral assignments/assessments

Methods of Instruction: Instruction will consist of lectures, laboratory demonstrations and assignments, and programming examples.

Outcomes Assessment: Exam questions are blueprinted to course objectives. Checklist rubrics are used to evaluate the projects for the presence of course objectives. Data is collected and analyzed to determine the level of student performance on these assessment instruments in regards to meeting course objectives. The results of this data analysis are used to guide necessary pedagogical and/or curricular revisions.

Course Requirements: All students are required to:
1. Maintain regular attendance and take part in class discussions.
2. Complete assigned homework and projects on time.
3. Take all exams as scheduled.

Methods of Evaluation: Final course grades will be computed as follows:

<table>
<thead>
<tr>
<th>Grading Components</th>
<th>% of final course grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework, class participation and attendance</td>
<td>10%</td>
</tr>
</tbody>
</table>

Students must practice skills on their own by doing homework to be able to master course objectives. Homework assignments relate to these objectives. Attendance and class participation are necessary for students to benefit from the guidance of the instructor.
Methods of Evaluation (continued):

<table>
<thead>
<tr>
<th>Grading Components</th>
<th>% of final course grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 9 or more Projects (dates specified by the instructor) Projects will show evidence of the extent to which students meet course objectives. Students should show that they have synthesized a combination of concepts. (See pages 6 and 7 for suggested projects.)</td>
<td>35%</td>
</tr>
<tr>
<td>• Midterm Exam</td>
<td>25%</td>
</tr>
<tr>
<td>The same objectives apply as with projects, but it is anticipated that students will show mastery of content covered in the first half of the course.</td>
<td></td>
</tr>
<tr>
<td>• Final Exam</td>
<td>30%</td>
</tr>
<tr>
<td>The same objectives apply as with the midterm exam, but it is anticipated that students will provide increased evidence of synthesizing a combination of concepts.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**: The above ‘% of final course grade’ guide is a suggestion and will vary depending on the instructor. The instructor must inform the students of the specific method of calculation on the class syllabus.

**Academic Integrity**: Dishonesty disrupts the search for truth that is inherent in the learning process and so devalues the purpose and the mission of the College. Academic dishonesty includes, but is not limited to, the following:

- plagiarism – the failure to acknowledge another writer’s words or ideas or to give proper credit to sources of information;
- cheating – knowingly obtaining or giving unauthorized information on any test/exam or any other academic assignment;
- interference – any interruption of the academic process that prevents others from the proper engagement in learning or teaching; and
- fraud – any act or instance of willful deceit or trickery.

Violations of academic integrity will be dealt with by imposing appropriate sanctions. Sanctions for acts of academic dishonesty could include the resubmission of an assignment, failure of the test/exam, failure in the course, probation, suspension from the College, and even expulsion from the College.

**Student Code of Conduct**: All students are expected to conduct themselves as responsible and considerate adults who respect the rights of others. Disruptive behavior will not be tolerated. All students are also expected to attend and be on time for all class meetings. No cell phones or similar electronic devices are permitted in class. Please refer to the Essex County College student handbook, *Lifeline*, for more specific information about the College’s Code of Conduct and attendance requirements.

Additional required course materials: USB flash memory and a note book for taking notes.

<table>
<thead>
<tr>
<th>Week (3 meetings @ 80 minutes)</th>
<th>Topic/Chapter</th>
</tr>
</thead>
</table>
| 1                             | User-Defined Simple Data Types (Chapter 8)  
Namespaces  
The String Type |
| 2                             | Arrays and Strings (Chapter 9) |
| 3                             | Using Records (Chapter 10)  
Introduction to Records  
Computing with String Objects  
The istream and ostream Classes |
| 4                             | Building Classes (Chapter 11)  
Designing a Class  
Implementing Class Attributes  
Implementing Class Operations |
| 5                             | Files and Streams (Chapters 3 & 9)  
ifstream and ofstream Objects  
String Streams  
Objects and File I/O |
| 6                             | Pointers, Run-Time Allocation, and Linked Lists (Chapter 13)  
Introduction to Pointer Variables  
Run-Time Allocation Using New and Delete  
Introduction to Linked Lists |
| 7                             | Overloading and Templates (Chapter 14)  
Recursion (Chapter 16) |
| 8                             | **Midterm Exam** |
| 9                             | Linked Lists (Chapter 17)  
ADT  
Unordered Linked Lists  
Ordered Linked Lists  
Doubly Linked Lists |
<table>
<thead>
<tr>
<th>Week (3 meetings @ 80 minutes)</th>
<th>Topic/Chapter</th>
</tr>
</thead>
</table>
| 10 | Stacks and Queues (Chapter 18)  
Implementation of Stacks as Arrays  
Linked implementation of Stacks  
Application of Stacks  
Queues  
Application of Queues |
| 11 | Searching and Sorting (Chapter 19)  
Search Algorithms  
Asymptotic Notation: Big-O Notation  
Sorting Algorithms |
| 12 | Binary Trees (Chapter 20)  
Searching |
| 13 | Arrays, vector <T>s, STL (Chapter 22)  
The vector <T> Class Template  
The Standard Template Library |
| 14 | Multidimensional vector <T> Objects (Chapter 22)  
Two-dimensional vector <T> Objects  
Two-dimensional vector <T> Operations |
| 15 | Data Structures (Chapter 15)  
The Stack Container  
The Queue <T> Adapter  
dequeue <T> Container  
Introduction to Trees |

**Final Exam**

**NOTE:** 9 or more projects are assigned on an ongoing basis throughout the semester to correspond to the topics being discussed in class. Roughly one project is due each week (with the exception of exam weeks) beginning in the third week of the semester.
CSC 122 Suggested Projects

Guidelines:

1. All programs must be documented. The student’s name must appear at the top of the source code and every other file.
2. The project number and the date the project is due and the date it is handed in must be at the top of the documentation for the source code.
3. The student must explain what the program does, how it does it and what the known limitations are (if any exist).
4. In cases where there are library files, these should also be documented.
5. The C++ code should also be documented. The student should have a comment for every two to three lines of code.
6. If a student decides to complete a program that is different from the given assignment, the student must explain that in the documentation of the program.

Suggested Assignments:  **NOTE:** Suggested projects listed below are illustrative of assignments that are distributed through the course to correspond to the topics being discussed in class.

**Project 1:** Print out the source code and the results of the execution of a program prepared for CSC 121.

**Project 2:** Complete and modify the code for the game “Rock, Paper Scissors” so that the player plays against the computer (use a random generator to generate the computer’s play) and there is a record that is kept on file of the game. The record should be created if the player is playing for the first time. If the player has already played before, her/his record should be loaded into the computer and the competition continued from there. Remember to document the program explaining what the code does and how it has been modified. The purpose of this assignment is to begin to understand how computers can keep records and analyze the record and modify its “behavior” using stored data. It also gives the student some understanding of the issues of trying to get a machine to produce ‘random’ responses.

**Project 3:** This project requires the student to create a record (in C++ called a **struct**). The record should contain several fields of different data types and should allow the user to search the records to find the student with the highest grade. In this project the student should gain an understanding of the nature of records with multiple data types, how to save the records, search the records and retrieve them. The special problems of maintaining records of multiple data types on a digital machine will be seen.

**Project 4:** Complete the program to use the “Clock class” and modify the class by overloading a new operator. Document the code and the changes that have been made and test the changes in the execution of the program. This program should give the student an understanding of some key issues in Computer Science: information hiding and overloading.

**Project 5:** Create a linked list and print the list forward and backward. Document the code and make sure to include an explanation of how the link list has been created. If the student would like to add an additional feature, such as a search function, he/she may feel free to do so. This assignment deals with dynamic memory allocation – its limitations and possibilities – and issues of memory addressing, memory storage and memory accessing.
Project 6: Create a double-linked list and traverse the list forward and backward. Document the code and make sure to include an explanation of how the link list works. If the student adds additional features, he/she should document that as well. In testing the list, show what happens in the case of the NULL list. Show why this type list is easier to search than a single-linked list.

Project 7: Create a program that uses two different methods to sort a list and compare the efficiency of each way of sorting the list. Document the code and explain how the efficiency of the algorithm is measured. Document the output to show what is found.

Project 8: Create an ordered binary tree. Print out the tree. Search on the tree and show the efficiency of the search. Document the code and the output.

Project 9: Use the library to find books on a topic in computer science, which is assigned by the instructor. The student will report on the books, giving their bibliographical information and assessing their usefulness for information.